16-741: Mechanics of Manipulation
Project Report

Object Pose Estimation from Manipulator Pose

Angelos Mavrogiannis (angelosm)
Satyaki Chakraborty (schakra1)
Suman Pokharel (spokhare)

12/10/2018

1. Introduction

While manipulating objects in activities of daily living, we come across a problem where objects are quite often severely occluded from the egocentric viewpoint making it difficult for the object of interest to be tracked. Most of the current approaches rely heavily on either the visual cues of the object of interest or its prior shape or both. While these approaches perform extremely well for low level of occlusion or mild occlusion, given enough data for object's prior shape, the dependency of these methods on visual cues and object priors creates a big challenge while dealing with objects under severe occlusion or complete occlusion. Our project aims to find out whether there exists a correlation between the manipulator pose and the object pose. With our approach, we aim to solve the problem of tracking severely occluded objects solely based on the manipulator pose.

2. Related Work

One of the approaches [1] proposes a spatial-temporal memory network (STMN) for video object detection which considers objects under low to severe occlusion. It uses recurrent computational unit to integrated pre-trained image classification weights in memory and in-network alignment module that spatially aligns the memory across time. This approach relies on the visual cues of the object from the past frames in sequence to estimate the object pose in current frame. This can be very useful even for severely occluded images given there is enough temporal information of the object and the object is occluded only for short temporal instances. However, the fact that it heavily relies on the temporal data severely limits its applicability for the cases where the object of interest is occluded for longer instances or when there is not enough temporal data of the object.

Another approach [2] implements a deep learning network framework that combines tubelet proposal generation and temporal classification with visual-temporal object cues. The algorithm captures the spatiotemporal locations of the object in videos and uses temporal CNN long short-term memory (LSTM) model in order to classify the tubelet proposals with both spatial and temporal features. This object motion model can be used to extrapolate the object pose at an occluded instance. This approach takes longer spatial-temporal feature into consideration, however, still heavily relies in the unoccluded visual cues of the object at a different segment of temporal instances.

Another approach described in [3] is based on shape-based template matching, which can be applied on isolated objects as well as objects in cluttered images and involves shape representation and estimation of similarity between the object pairs. This approach requires a prior of the object's shape for all possible orientations and generates a template for the object. It then generates a similar model of the visual cues of the object at an occluded instance and tries to match the shape of the object at a low-level gradient without extracting contours. While this approach works well even with untextured object under occlusion, it still performs poorly for occlusion that is greater than 35%.

Our key observation was that moving objects or objects of interest in activities of daily living under manipulation suffer occlusion because of hand interaction or dexterous manipulation and we are interested in finding whether we can use the information from the manipulator, for instance hand pose, to estimate the object pose to track the object. Thereof, our project proposes an alternative approach where we use only the manipulator pose to estimate the object pose for object tracking in severely occluded environments.

3. Problem statement formulation

3.1 Choosing environment (2D vs 3D)

While the problem that we are trying to address is present in 2D scenes (since occlusion happens due to projection of 3D points from a certain viewpoint), we could not find any 2D dataset suitable for our task. Most 2D datasets for object tracking from egocentric viewpoints have only object annotations. For tracking occluded objects with hand poses as cues, datasets need to contain both hand and object annotations. We only found one dataset relevant for such task. [6] However the dataset is significantly small, and this makes it hard to train neural network models without overfitting to the seen examples. This poses a new challenging problem of lack of annotated data for joint pose and object tracking. We circumvent this problem by choosing a 3D simulator, where we perform hand manipulation of different objects. Since, our goal is to reliably track pose of an object under dexterous manipulation, being able to do that in a 3D simulated environment implies that other than visual cues of the object, hand pose can be used as a significant input for tracking objects in 2D egocentric scenes, which was the main motivation for this project.

3.2 Problem statement definition

Now that we have mentioned verbally the problem statement, we are going to define it mathematically in this subsection.

Let p^t_{man} and p^t_{obj} denote manipulator poses and object poses at time instant t respectively. Our goal is to verify whether these two entities are highly correlated and whether you can estimate p^t_{obj} from p^t_{man}.

One observation we made is that the current pose of the object is dependent on 3 factors: 1) the previous position of the object 2) the manipulator joint positions 3) manipulator joint motion. Directly estimating p^t_{obj} from p^t_{man} can be difficult as that would require severe data augmentation for all possible previous poses of the object. Instead we try to estimate the change in the object pose $(\Delta p_{obj}^{t-1:t})$ from the change in the manipulator pose ($\Delta p_{man}^{t-1:t}$), previous configuration of the robot joints (p_{man}^{t-1}) and previous pose of the object (p_{obj}^{t-1}). So, the end goal is to be able to learn to approximate a regression function f given by

$$\Delta p_{obj}^{t-1:t} = f(\Delta p_{man}^{t-1:t}, p_{man}^{t-1}, p_{obj}^{t-1}) \qquad (1)$$

And the current object pose is estimated by adding the delta to our previous estimate of the object pose. Thus, we have formulated our problem as a pose tracking problem. The next section is going to describe what the inputs signify physically in the simulator.

4. Experimental Setup

4.1 Manipulator Arm

The manipulator arm we decided to use in our simulation is the Shadow Dexterous Hand (Smart Motor actuation system version), which incorporates anthropomorphic features in its design and gives a very close approximation to the real human hand. It has 24 joints, four of which are coupled and hence it can offer 20 actuated degrees of freedom, that can be controlled to ± 1 degree across the full range of movement, and 4 underactuated degrees of freedom. More specifically, the thumb has five joints and five degrees of freedom, while each of the other fingers has four joints and three degrees of freedom, except for the little finger. The little finger has an extra revolute joint on the palm of the hand, which allows opposition to the thumb. Finally, the wrist has 2 degrees of freedom, as it can pitch and yaw, as shown in Figure 1. [4]

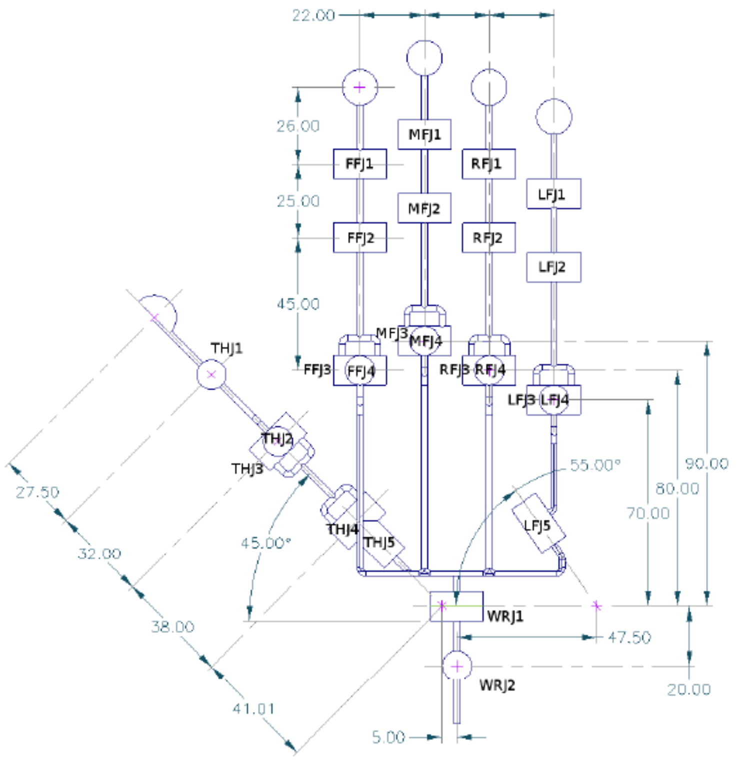


Figure 1: Kinematics diagram of the shadow hand

Some of the anthropomorphic features of the hand are the carefully measured distances between the fingertips and the hand knuckles, as well as the design constraint that allows the angle of the middle joint of each finger to only be equal or greater than the angle of the distal joint of this finger. These characteristics are crucial for making the shadow dexterous hand simulate the human hand in a realistic way.

4.2 Simulation Environment

As for the simulation software, we decided that the OpenAI Gym would be good for developing as well as comparing Reinforcement Learning algorithms. The default simulator, developed by OpenAI, initially sets a goal orientation for an egg-shaped object and then controls the shadow hand actuators and changes the values of the joint angles during each iteration until the object reaches its target orientation as can be seen in Figure 2. [5]

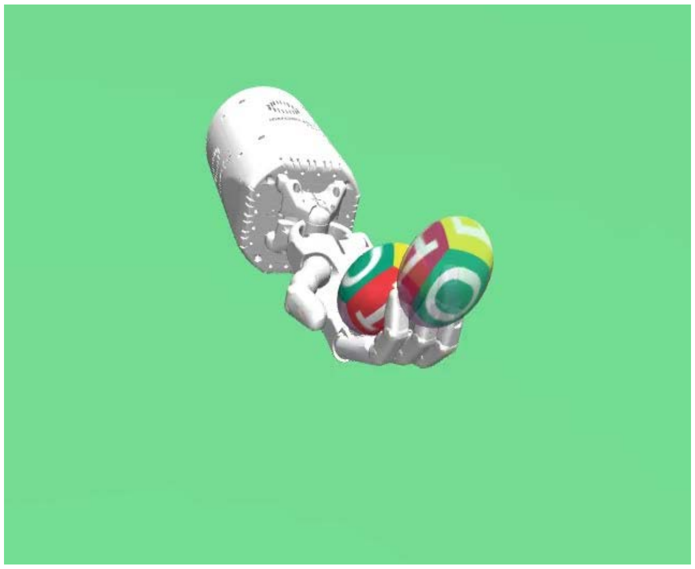
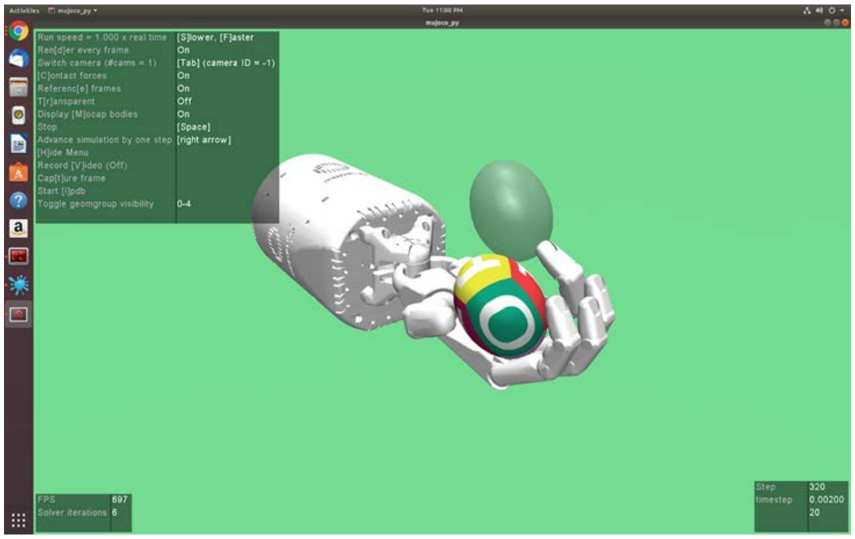


Figure 2: Default gym simulator for manipulating an egg-shaped object

We modified the simulator in order to demonstrate the estimation of the pose of the object. The estimated pose is represented by the hollow green egg-shaped object in our simulator as shown in Figure 3.



```
Run speed = 1.000 x real time   [S]lower, [F]aster
Ren[d]er every frame            On
Switch camera (#cams = 1)       [Tab] (camera ID = -1)
[C]ontact forces                On
Referenc[e] frames              On
T[r]ansparent                   Off
Display [M]ocap bodies          On
Stop                            [Space]
Advance simulation by one step  [right arrow]
[H]ide Menu
Record [V]ideo (Off)
Cap[t]ure frame
Start [i]pdb
Toggle geomgroup visibility     0-4
```

```
FPS                697
Solver iterations  6
```

```
Step      320
timestep  0.00200
          20
```

Figure 3: Our modified simulator that demonstrates the estimated pose of the object

As illustrated in Figure 4., our learning objective is to take a 24-dimensional vector describing the joint angles of the manipulator and learn a function in order to estimate the pose of the egg-shaped object, which is described by a 7-dimensional vector that consists of the 4-dimensional quaternion and the position of the object.

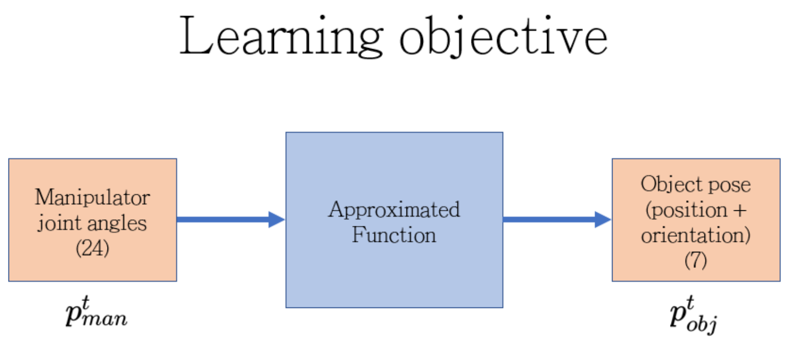

Learning objective
Manipulator joint angles (24)
Approximated Function
Object pose (position + orientation) (7)
p^t_{man}
p^t_{obj}

Figure 4: The learning objective of our method

5. Experiments

5.1 Choosing a learning model

We initially tried to keep the model simple by assuming the function f is a linear regressor or a polynomial regressor (up to 5th degree). A large generalization error during test time implied that the model was underfitting. So, we finally used a 3-layer MLP with tanh nonlinearity as our learnable module. Figure 5. shows the architecture.

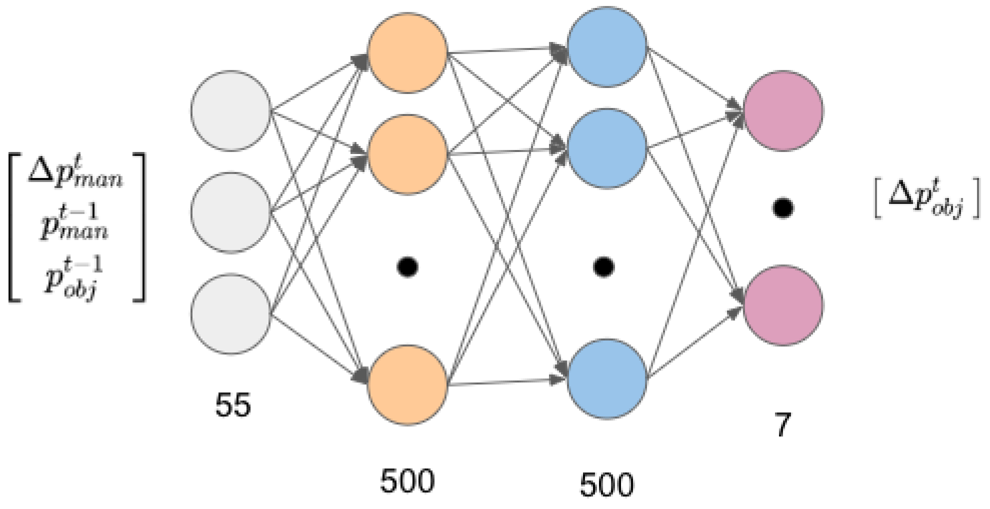

$\begin{bmatrix} \Delta p^t_{man} \\ p^{t-1}_{man} \\ p^{t-1}_{obj} \end{bmatrix}$
55
500
500
7
$[\Delta p^t_{obj}]$

Figure 5: The architecture of the model

The input to our model is a 55-dimensional vector which is a concatenated vector of manipulator pose change, previous manipulator poses, previous object poses. Next there are 2 hidden layers each with 500 units and tanh nonlinearity and the final layer has 7 units with no non-linearity since it estimates the delta corresponding to object pose. The 7 dimensions correspond to the 3 positional coordinates in space and the quaternion.

5.2 Choosing a loss function

Our initial set of experiments with an L2 loss saw relatively unstable convergence during training time. This can be attributed due to gradient explosion, instead of clipping the gradients beyond a certain threshold, we used a smooth L1 loss function. The Smooth L1 loss function is piecewise L1 and L2 as shown in Eq. (2) and Figure 6. It is an L1 loss beyond |x| >=1 and L2 loss in the range |x| < 1. This loss is less susceptible to gradient explosion and converges more smoothly for unbounded regression values.

$$smooth_{LI_{plot}} := piecewise(abs(x) < 1, 0.5 \cdot x^2, abs(x) - 0.5) \qquad (2)$$

$$\begin{cases} 0.5\,x^2 & |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

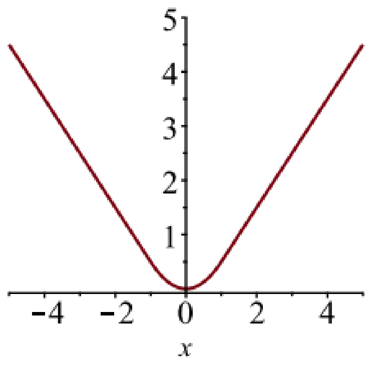


Figure 6: Smooth L1 loss

We took the original version defined in the Fast RCNN paper [7]. The more generic version of the smooth L1 loss is shown in [8]. It uses another term σ to account for the unnormalized regression values. We use $\sigma = 0.3$ for all our experiments.

5.3 Choosing an object for manipulation

As explained in the previous section, we are using the OpenAI gym environment. The simulator with the shadow robot arm has manipulation options for 3 different type of objects: cube, egg(ellipsoid) and stick [5] as shown in Figure 7&8.

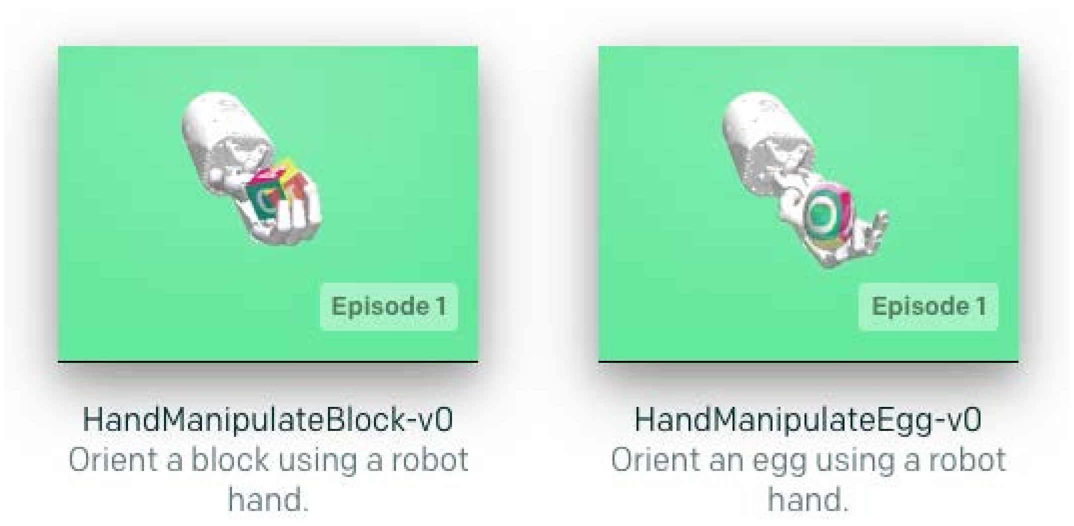

Episode 1
Episode 1
HandManipulateBlock-v0
Orient a block using a robot hand.
HandManipulateEgg-v0
Orient an egg using a robot hand.

Figure 7: Cube and Egg-shaped object manipulation on OpenAI Gym

Figure 8: Manipulation of a pen on OpenAI Gym

We choose the second one (ellipsoid) because it seemed to be the most representative of generic objects under manipulation amongst all the three. Predicting the orientation of the ellipsoid is also going to be more challenging than that of the cube since a cube is more likely to be in a stable orientation and less likely to roll.

5.4 Choosing manipulation configuration

We never fully grasp (there is no force closure) the object under manipulation as that is a less interesting case. Instead we loosely hold the object under manipulation and make sure that the object never really falls off the palm of the manipulator. We bound the initialization constraints of the joint angles to make sure the manipulator is holding the object. Next, we run 10,000 iterations in each of which we alter the joint angles randomly within a bound of 1 radian. We collect the data for a couple of hours with approximately 10 million ordered pairs of manipulator poses and object poses. (Our model is trained in a single 1080Ti GPU with a batch size of 128. The simplicity of the model enables training within a few hours and gives real-time performance during test time on a standard CPU.)

6. Experiments

6.1 Tracking results (demo)

We show the tracking results with our method for 5k (Figure 9,10&11) and training 100k (Figure 12,13&14) iterations respectively. At every time instant ‘t’ we evaluate $\Delta p_{obj}^{t-1:t}$ from Eq. (1) and estimate current object pose as

$$p_{obj}^{t} = p_{obj}^{t-1} + \Delta p_{obj}^{t-1:t} \qquad (3)$$

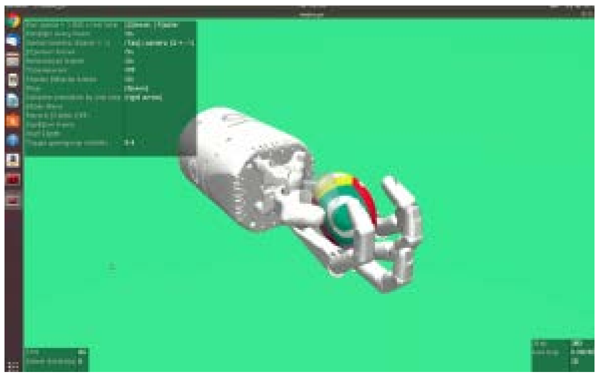


Figure 9: 5k training iterations
Test time demo at t=10
Estimated pose (shown in grey ellipsoid) starts to drift

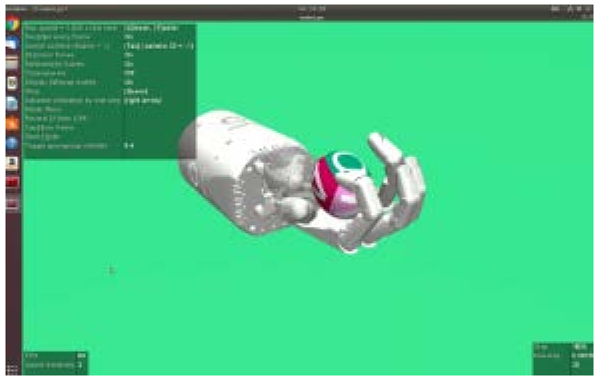


Figure 10: 5k training iterations
Test time demo at t=50
Estimated pose is way off

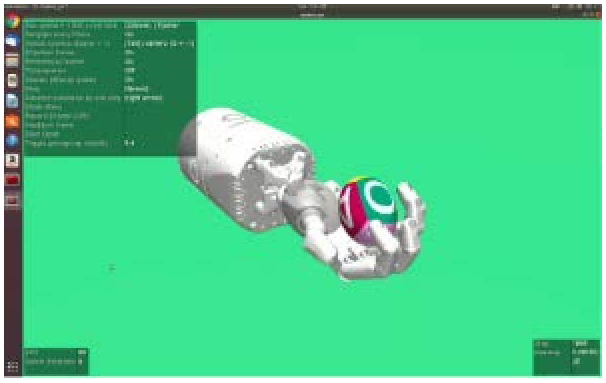


Figure 11: 5k training iterations
Test time demo at t=100
Estimated pose is way off

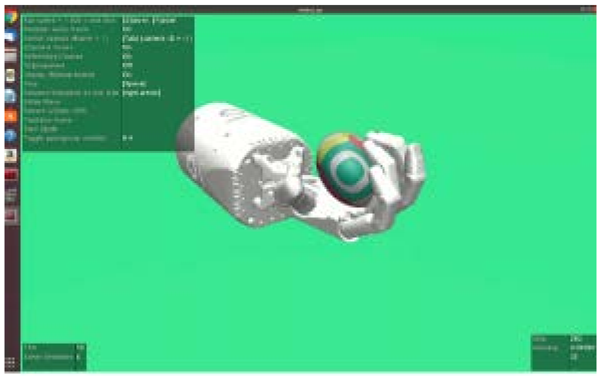


Figure 12: 100k training iterations
Test time demo at t=10
Estimated pose overlaps almost completely

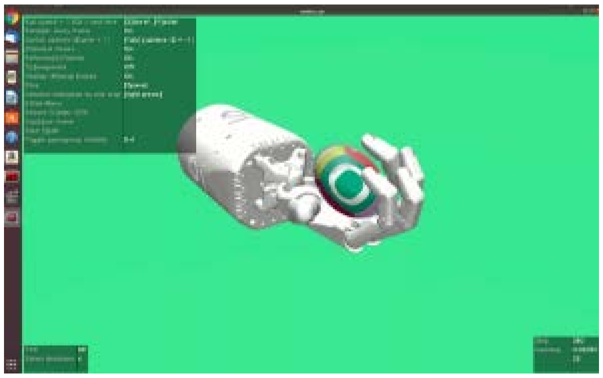


Figure 13: 100k training iterations
Test time demo at t=50
Drift is hard to notice

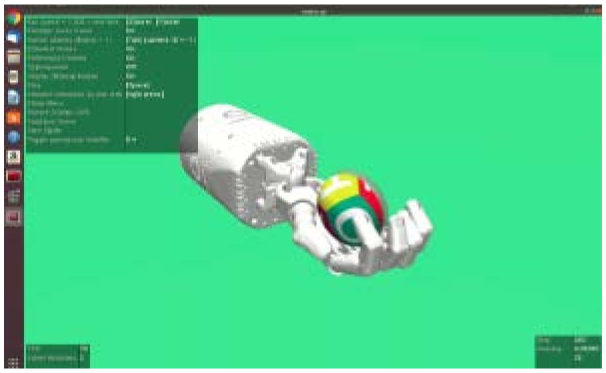


Figure 14: 100k training iterations
Test time demo at t=100
Drift is present but negligible

As we can see in the above Figures 9-14, the network learns to track reliably well after 100k training iterations. This can be attributed due to the low drift during test time. The following graph shows the test time mean squared error for different training checkpoints.

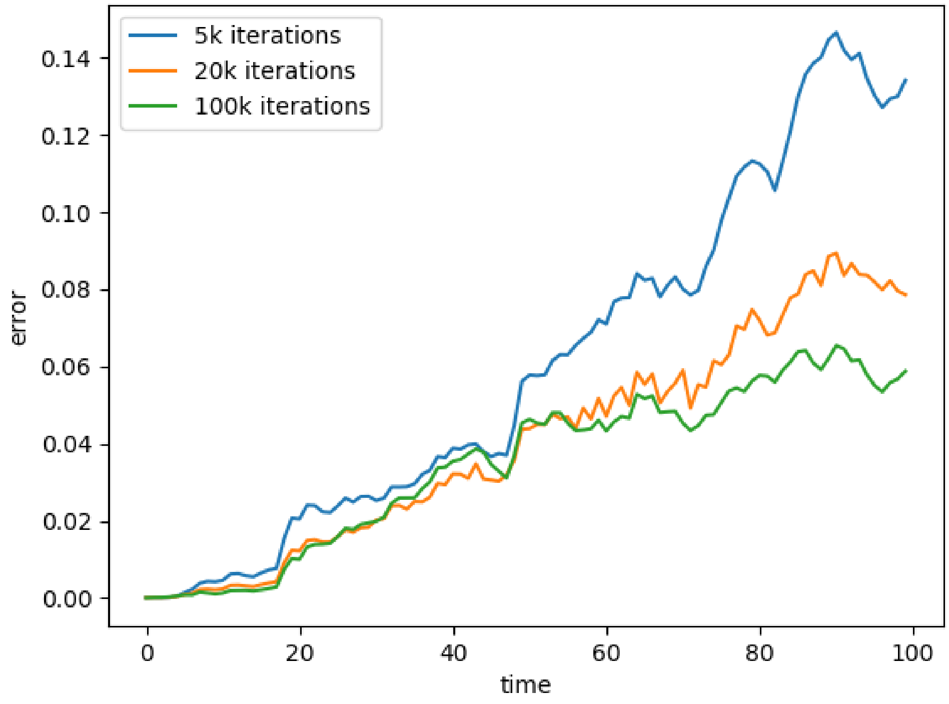


Figure 15: Test time error and drift

As is shown in Figure 15, the error curve becomes more and more flat as the number of training iterations increases. Also, to be noted is the fact that with time, for a given checkpoint, the error increases. This test time error is the main cause for drift in tracking. This can be explained from Eq. (3). Unrolling Eq. (3) in time gives:

$$p^t_{obj} = p^0_{obj} + \Delta p^1_{obj} + .. + + \Delta p^t_{obj} \qquad (4)$$

This means our current estimate of the object pose is the initial object pose + sum of all the deltas of object pose predicted over time. During test time, there will always be some generalization error and this error will accumulate over time to cause drift.

7. Conclusion and future work

- With these set of experiments, we show that the hand poses and manipulated object poses are highly correlated and we present a learnable module which can estimate the object pose solely from the data distribution.
- Our approach uses a very simple neural network to model this and thus achieves real time performance. It takes 62ms on a standard CPU machine compared to the 280ms computation time for the simulation update step.
- However, the main goal of the project is to solve the case for 2D objects, because occlusion is the main problem for such datasets. This means when enough visual cues of the object are not present, we can use hand poses (may be from pose machines) as additional input to any model to track objects more reliably. Next, we present an FQA section where we discuss the most probable questions that can arise from our work.

8. FQA

How to reduce/prevent drift?

The drift issue is something very fundamental to a tracking problem and it happens due to the error accumulation as explained in the results section. It can be reduced with more training but cannot be prevented.

Does the model not learn biases for object shapes?

Yes, it does. OpenAI gym only comes with 3 different object shapes and we took the most representative/challenging one for our model. With enough data we can probably pass the shape parameters as an input to the function to take this issue into account.

Is the model not just learning forward kinematics?

No. The model would have learnt forward kinematics if the object was under full grasp, then that object could have been treated as an end actuator. But such is not the case when the object is loosely held and under dexterous manipulation.

References

[1] Xiao, Fanyi, and Yong Jae Lee. "Video Object Detection with an Aligned Spatial-Temporal Memory." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[2] Kang, Kai, et al. "Object detection in videos with tubelet proposal networks." *Proc. CVPR*. Vol. 2. No. 6. 2017.

[3] Hsiao, Edward, and Martial Hebert. "Gradient Networks: Explicit Shape Matching Without Extracting Edges." *AAAI*. 2013.

[4] Walker, R. "Shadow dextrous hand technical specification." *Shadow Robot Company* (2005).

[5] https://gym.openai.com/envs/#robotics

[6]Sridhar, Srinath, et al. "Real-time joint tracking of a hand manipulating an object from rgb-d input." *European Conference on Computer Vision*. Springer, Cham, 2016.

[7] Girshick, Ross. "Fast r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2015.

[8] https://github.com/rbgirshick/py-faster-rcnn/files/764206/SmoothL1Loss.1.pdf